SAD 371 Assignment 3

Date: 07/10/2025

TEAM MEMBERS: JESSICA WIEHE 600242, KAELO MOLEBATSI 600697, ARNAN KRUGER 601326, HENDRIK SMITH 600519, MARIUS VAN DER WESTHUIZEN 601075

Table of Contents

[1. Acquisition Strategy (5 Marks) 2](#_Toc211004610)

[2. Architecture Design (5 Marks) 3](#_Toc211004611)

[3. Hardware and Software Design (5 Marks) 5](#_Toc211004612)

[Class and Method Design (15 marks) 6](#_Toc211004613)

[Physical Data Models (Refined ERD) (15 marks) 6](#_Toc211004614)

[Interface Design (10 marks) 6](#_Toc211004615)

[References 7](#_Toc211004616)

# 1. Acquisition Strategy (5 Marks)

The buy-or-make or outsourcing choice is what determines if a system is to be developed in-house, purchased as an off-the-shelf solution, or outsourced to have another person develop it. The decision influences cost structure, human resource needs, knowledge retention, and long-term intellectual property ownership. Getting it right ensures that the system aligns with strategic goals as well as remains functional (Umbrex, 2025)

The selection process utilizes the core-vs-context test to separate activities that create competitive advantage (core) from those that are supportive or context (Umbrex, 2025). Core activities should be retained in-house, while context activities may be considered for outsourcing or off-the-shelf solution. Four diagnostic lenses are used to narrow down the choice: differentiation, competency, control and risk, and economics (Umbrex, 2025)

Differentiation Lens: Examines whether the system provides unique value to improve organizational competitiveness. If the system directly contributes towards performance or reputation, it will have to be developed in-house (Umbrex, 2025)

Competency Lens: Checks whether internal teams have the technical ability to deliver quality results or whether they can get that done within a specified time frame. Good quality internal capacity prefer in-house development (Umbrex, 2025).

Control and Risk Lens: Considers potential risks relating to data protection, regulation, and supply chain stability. Systems handling sensitive information should not be outsourced (Umbrex, 2025).

Economic Lens: Weighs total cost of ownership, including hardware, labour, and maintenance. The lowest cost option is only preferable if it does not compromise strategic or security requirements (Umbrex, 2025).

Each of the criteria is rated 1–5, and then weighted according to its strategic importance (Umbrex, 2025).

| **Criteria** | **Weight** | **Make** | **Buy** | **Outsource** |
| --- | --- | --- | --- | --- |
| Differentiation | 40% | 5 | 3 | 2 |
| Competency | 25% | 4 | 3 | 3 |
| Control & Risk | 20% | 5 | 3 | 2 |
| Economics | 15% | 4 | 5 | 4 |
| **Weighted Total** | 100% | **4.45** | 3.55 | 2.65 |

The best score is achieved by the Make choice. Developing the system internally ensures alignment with organizational goals, saves intellectual capital, and maintains control of confidential data (Umbrex, 2025). It also develops in-house technical expertise and allows for customization to specific specifications (Umbrex, 2025). Although buying or outsourcing may appear to be faster or cheaper on the surface, they are associated with long-term risks such as reduced flexibility, exposure of information, and dependency on external suppliers (Umbrex, 2025). Therefore, development in-house is the most sustainable and strategic way of acquiring (Umbrex, 2025).

# 2. Architecture Design (5 Marks)

Chosen Architecture: Three-Tier Client-Server Architecture (Cardoso, 2024)

Tier Descriptions:

* Presentation Tier: Handles all user input, i.e., web pages, mobile clients, or desktop clients. Displays data and provides input (Cardoso, 2024)
* Application Tier (Business Logic / Logic Tier): Converts user requests, executes business logic, and communicates with the database (Cardoso, 2024)
* Database Tier: Stores and manages persistent data. Can utilize relational or NoSQL databases depending on requirements (Cardoso, 2024).

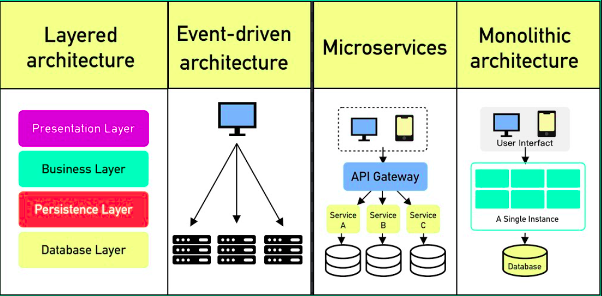
Rationale

* Separation of Concerns: Each layer has a particular responsibility, and so changes in one layer won't influence others. This is simple to manage (Cardoso, 2024).
* Scalability: Tiers are separate to be scaled. The database, for example, might be scaled to handle bigger data while the presentation tier is not (Cardoso, 2024).
* Reusability: Modules at each level can be reused in other modules or applications. The business logic level can have multiple interfaces (Cardoso, 2024).
* Performance and Reliability: Processing and data management are centralized in the application and database layers, reducing errors and improving stability (Cardoso, 2024).
* Real-World Fit: Suitable for applications like e-commerce websites, student welfare management systems, or other medium-complexity applications where users are addressed by a UI, logic is computed at a middle layer, and data is stored persistently (Cardoso, 2024).

Alternative Architectures Considered:

* Monolithic Architecture: Simple and fast to deploy for small systems but hard to scale and maintain as complexity grows (Cardoso, 2024).
* Microservices Architecture: Highly scalable and fault-tolerant but hard to deploy and manage due to service communication (Cardoso, 2024).
* Event-Driven / Saga / CQRS: Maintains asynchronous operations and distributed transactions but introduces complexity to the system and requires advanced monitoring (Cardoso, 2024).

The three-tier architecture keeps simplicity, maintainability, and scalability in equilibrium. It is well suited for multi-user applications with permanent data and future-proof to evolve in the direction of more advanced architecture if necessary (Cardoso, 2024).



Source: (Cardoso, 2024).

# 3. Hardware and Software Design (5 Marks)

**Hardware Requirements:**

| **Component** | **Specification** |
| --- | --- |
| Server | Quad-Core Processor (2.4GHz or higher) |
| Memory | 16GB RAM minimum |
| Storage | 500GB SSD |
| Network | Gigabit Ethernet with secure VPN |
| Backup Server | Redundant daily backup storage (off-site or cloud-based) |

**Software Requirements:**

| **Component** | **Technology / Tool** |
| --- | --- |
| Operating System | Linux (Ubuntu or CentOS) |
| Backend Framework | Node.js or Java (Spring Boot) |
| Frontend | HTML5, CSS3, JavaScript |
| Database | PostgreSQL |
| Authentication | Belgium Campus SSO Integration |
| Notification Service | SMTP + SMS Gateway |
| Version Control | GitHub |
| Development Tools | Visual Studio Code or IntelliJ IDEA |

**Justification:**

* **Scalability:** Node.js and PostgreSQL handle high concurrency efficiently, allowing hundreds of users to book simultaneously.
* **Cost:** Open-source tools lower licensing and maintenance costs.
* **Compatibility:** The stack aligns with Belgium Campus’s existing infrastructure.
* **Reliability:** Daily automated backups, encrypted communication, and secure login ensure data integrity and availability.

# Class and Method Design (15 marks)

# Physical Data Models (Refined ERD) (15 marks)

# Interface Design (10 marks)

# References

Cardoso, M., 2024. *Software architecture patterns overview.* [Online]   
Available at: https://www.linkedin.com/pulse/software-architecture-patterns-overview-mario-cardoso-ghwyf/

Umbrex, 2025. *Make-vs-buy / outsourcing decision.* [Online]   
Available at: https://umbrex.com/resources/strategic-sourcing-playbook/make-vs-buy-outsourcing-decision/